VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce mrkání očí v obrazech

Eye Blink Detection in Videos

2016 Adam Holomek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Adam Holomek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Detekce mrkání očí v obrazech
Eye Blink Detection in Videos

Jazyk vypracování: čeština

Zásady pro vypracování:

S rozvojem kamerových systémů se v posledních letech zvýšilo i jejich využití v oblasti detekce a analýzy různých částí lidských tváří (očí, rtů, duhovek). Detekce těchto částí za pomocí kamerových systémů může být využívána například v automobilech. Ty jsou již často vybavovány kamerovými systémy pro analýzu únavy řidiče. Cílem této práce bude vytvoření detektoru mrkání očí a otestování vytvořeného řešení.

1. Seznamte se se základními pojmy v oblasti detekce objektů v obrazech. Zejména se zaměřte na detekci očí v obrazech.
2. Seznamte se s knihovnou OpenCV.
3. S pomocí knihovny OpenCV vytvořte program, který bude detekovat lidské oči a rozpoznávat jestli jsou oči otevřené nebo zavřené.
4. Experimentálně ověřte funkčnost, přesnost a rychlost vytvořeného řešení.
5. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

[1] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I-511-I-518 vol.1 (2001)
[2] W.O. Lee, E.C. Lee, K.R. Park: Blink detection robust to various facial poses
Journal of Neuroscience Methods, pp. 356–372 (2010)
[3] Song F, Tan X, Liu X et al.: Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. Pattern Recogn. 47(9), pp. 2825-2838 (2014)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 1. dubna 2016

.....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této bakalářské práce je vytvoření detektoru, který bude snímat mrkání očí v obrazech. Jedná se především o vyhodnocení stavu, zda jsou oči otevřené nebo zavřené. Implementace tohoto detektoru proběhne třemi různými metodami, a to za pomoci opensource knihovny OpenCV. Součástí práce bude seznámení s těmito metodami, otestování funkčnosti a zhodnocení jejich úspěšnosti.

Klíčová slova: OpenCV, Viola-Jones, Houghova transformace, Detekce tváře, Detekce očí, Mrkání

Abstract

The goal of this thesis is to create a detector that will capture the eye blinking in videos. It is all about evaluation of the state, whether eyes are open or closed. Implementation of this detector will be held by three different methods and with the help of open source OpenCV library. Part of the work will be familiar with these methods, test the functionality and evaluate their success.

Key Words: OpenCV, Viola-Jones, Hough Transform, Face Detection, Eye Detection, Blinking

Obsah

Seznam obrázků

1 Úvod

Ospalost je jedním z hlavních důvodů vzniku dopravních nehod. Tyto nehody se vyznačují vysokou mírou úmrtnosti, což je zapříčiněno snížením pozorovacích schopností nebo schopnosti řídit. Mikrospánek trvá zhruba 2-3 sekundy a je dobrým identifikátorem únavy řidiče. Na trhu existuje celá řada bezpečnostních prvků, které jsou využívány k ochraně řidiče a posádky, například bezpečnostní pásy, airbagy, karosérie z pevného plechu a jiné. Tyto bezpečností prvky jsou však použity až v době samotné nehody. Vývoj technologií pro detekci nebo prevenci ospalosti za volantem je hlavní výzvou v oblasti systému pro předcházení dopravních nehod. Na základě sledování očí se předpokládá, že lze detekovat příznaky únavy řidiče v dostatečném předstihu tak, aby se zabránilo případné autonehodě. Detekce únavy zahrnuje sekvenci několika snímků tváře a pozorování očních pohybů.

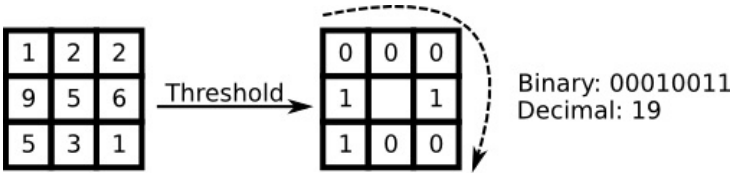
Tato práce se zabývá detekcí mrkání očí za použití open-source knihovny OpenCV, samotná implementace systému proběhne v jazyce C++. Při projektování systému bude kladen důraz na sledování přesného stavu očí, zda jsou otevřené nebo zavřené, a to i v reálném čase.

Postupně budou v této práci probrány metody, které povedou k detekci mrknutí oka. Představen bude postup tréninku kaskádového klasifikátoru, který bude použit jako jeden z metod detekce. Závěrem budou tyto použité metody otestovány, jejich přesnost detekce porovnána a zhodnocena.

2 Local Binary Patterns

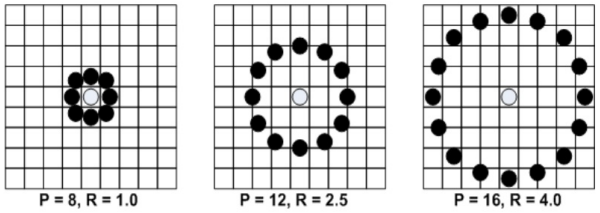
Local Binary Patterns (LBP) je neparametrický deskriptor, který efektivně shrnuje místní struktury obrazů. V posledních letech, kdy vzrostl zájem v mnoha oblastech zpracování obrazu a počítačového vidění, prokázal svou účinnost v celé řadě aplikací. Zejména tedy v obličejové analýze obrazu, která zahrnuje detekci obličeje, rozpoznávání obličejů a jiné.

Původní LBP operátor představil Ojala [1] v roce 1996 pro obrazový popis textur. Tento operátor pracuje s osmi okolními pixely, kde hodnota středového pixelu slouží jako prahová. Pokud má okolní pixel větší nebo stejnou hodnotu šedé jako samotný středový pixel, je tomuto okolnímu pixelu přiřazena hodnota 1, jinak 0. Vytvořením řetězce těchto osmi hodnot se dostane LBP kód středového pixelu, jak je znázorněno na obrázku 1.



Obrázek 1: Původní LBP operátor [2]

Později byl LBP operátor modifikován, což umožnilo práci s různým počtem okolních pixelů. V tom případě představuje P počet okolních pixelů, které jsou porovnávány se středovou hodnotou a R poloměr kruhu od středového pixelu (obrázek 2).



Obrázek 2: Varianty okolí středového pixelu pro různé hodnoty P a R [3]

Jestliže jsou souřadnice středového pixelu (x_c, y_c), pak souřadnice jeho P okolních pixelů (x_p, y_p), které se nacházejí na okraji kruhu s poloměrem R, jsou dány vztahem:

$$x_p = x_c + R\cos(\tfrac{2\Pi p}{P})$$
$$y_p = y_c + R\sin(\tfrac{2\Pi p}{P})$$

V případě, že hodnota šedé středního pixelu g_c a hodnoty šedé jeho okolních pixelů jsou g_p, s $p = 0...P-1$, pak textura T v lokálním okolí může být definována jako:

$$T = t(g_c, g_0, g_1, ..., g_{P-1})$$

Texturu T je možné popsat i jiným způsobem. Bez ztráty informací je odečtena hodnota středního pixelu g_c od hodnot okolních pixelů g_p

$$T = t(g_c, g_0 - g_c, g_1 - g_c, ..., g_{P-1} - g_c)$$

Za předpokladu, že jsou rozdíly nezávislé na g_c, bude vztah popsaný výše faktorizován

$$T \approx t(g_c)t(g_0 - g_c, g_1 - g_c, ..., g_{P-1} - g_c)$$

Člen $t(g_c)$ popisuje celkový jas obrazu, neobsahuje však žádné užitečné informace o textuře. Z tohoto faktu bude tento člen ignorován

$$T \approx t(g_0 - g_c, g_1 - g_c, ..., g_{P-1} - g_c)$$

Při změnách průměrné hodnoty šedé jsou značené rozdíly $g_p - g_c$ neměnné. Této neměnnosti, s ohledem na nastavení rozsahu šedé stupnice, lze dosáhnout tak, že budou považovány pouze náznaky rozdílu namísto jejich přesných hodnot

$$T \approx t(s(g_0 - g_c), s(g_1 - g_c), ..., s(g_{P-1} - g_c))$$

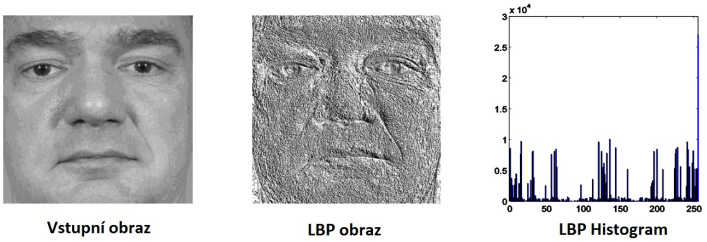
kde

$$s(x) = \begin{cases} 1 & \text{jestliže } x \geq 0 \\ 0 & \text{jinak} \end{cases}$$

K vytvoření LBP pro pixel (x_c, y_c) je potřeba přiřadit binomickou váhu 2^P ke každému znamení $s(g_P - g_c)$. Hodnota LBP kódu pixelu (x_c, y_c) je definována jako:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

Ze všech hodnot LBP je následně vytvořen histogram četnosti výskytu, tzn. příznak obrazu. Převod vstupního obrazu na obraz LBP, včetně histogramu četnosti výskytu, je znázorněn na obrázku 3.



Obrázek 3: Příklad LBP obrazu a histogramu [3]

2.1 Uniformní vzory

Pojem uniformní vzory představil Mäenpää v roce 2000 [4]. Bylo zjištěno, že určité vzory jsou základní vlastností textury. Tyto vzory se nazývají uniformní, protože mají jednu věc společnou, obsahují nanejvýš dva přechody z 1 na 0 nebo z 0 na 1 v binárním kódu. Definice uniformního vzoru zavádí tzv. uniformní hodnotu U, která obsahuje počet přechodů mezi 1 a 0. Vyjádřeno vztahem:

$$U(LBP_{P,R}) =\mid s(g_{p-1} - g_c) - s(g_0 - g_c) \mid + \sum_{p=1}^{P-1} \mid s((g_p - g_c) - s(g_{p-1} - g_c) \mid$$

Vzory s hodnotou U nižší nebo rovnou dvěma jsou označovány jako uniformní. Uniformní vzory umožňují vidět metodu LBP jako unifikovaný přístup k tradičně rozdílným statickým a strukturálním modelům textury. Každý pixel je označen kódem textury primitiv vzhledem k jeho lokálnímu okolí. Takto může být každý LBP kód považován za micro-texton (mikrostruktura v obrazu buněk). Mezi místní primitiva zjištěné pomocí LBP patří body, roviny, konce čar, hrany, rohy a jiné.

2.2 Rotační invariance

Je-li potřeba odstranit efekt rotace, musí být každý LBP kód otočen zpět do jeho referenční polohy. Tuto transformaci je možné definovat následujícím vztahem:

$$LBP^{ri}_{P,R} = \min\{ROR(LBP_{P,R}i) \mid i = 0, ..., P-1\}$$

Horní index ri odpovídá využití rotační invariance a $ROR(x, i)$ vykonává kruhový bitový posun doprava i-krát o P-bitové číslo.

2.3 Uniformní rotačně invariantní vzory

Uniformní rotačně invariantní vzory vznikají kombinací principu invariance a uniformnosti. Tyto vzory jsou definovány pomocí následující rovnice:

$$LBP^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c)2^p & \text{jestliže } U(LBP_{P,R}) \leq 2 \\ \text{P+1} & \text{jinak} \end{cases}.$$

Horní index $riu2$ odpovídá využití uniformních rotačně invariantních vzorů, které mají hodnotu U. Tímto krokem se výrazně sníží počet vzorů na $P + 1$.

3 Objektový detektor Viola - Jones

Jedná se o detektor objektů, který pracuje s obrazem ve stupních šedi. Poprvé byl tento detektor představen v roce 2001 P. Violou a M. Jonesem [5]. Jeho hlavní výhodou je rychlost, nezávislost na intenzitě osvětlení či velikosti sledovaného objektu. Detektor Viola-Jones se skládá ze 4 klíčových částí: Integrální obraz, Haarovy příznaky, AdaBoost a Kaskadový klasifikátor.

3.1 Integrální obraz

Prvním krokem detektoru Viola-Jones je převedení vstupního obrazu na obraz integrální (obrázek 4).

1	1	1
1	1	1
1	1	1

1	2	3
2	4	6
3	6	9

Obrázek 4: Převod vstupního obrazu na obraz integrální

V takovém obrazu odpovídá každý bod o souřadnicích (x, y) součtu jasových hodnot pixelů nacházejících se vlevo a nad od souřadnic (x, y).

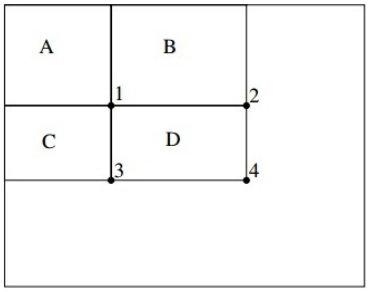
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

kde $ii(x, y)$ je integrální obraz a $i(x, y)$ je originální obraz. Dále jsou použity následující vztahy

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

kde $s(x, y)$ je kumulovaný součet hodnot řádku obrazu. Také platí, že $s(x, -1) = 0, ii(-1, y) = 0$. Pro příklad, obrázek 5 zobrazuje jednotlivé oblasti pro výpočet hodnoty integrálního obrazu v obdélníku D.

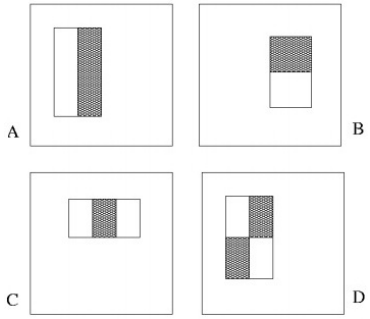


Obrázek 5: Zjištění hodnoty integrálního obrazu

Součtem pixelů v obdélníku A dostaneme hodnotu integrálního obrazu v bodě 1. Hodnota v bodě 2 je dána součtem pixelů v obdélnících A a B, v bodě 3 pak obdélníky A a C. Bod 4 je součet obdélníků A, B, C a D. Sumu uvnitř obdélníku D můžeme vyvodit ze vztahu: 4 - (2 + 3) + 1

3.2 Haarovy příznaky

Za Haarovy příznaky se považují obdélníkové regiony v určitě části okna detekce. Každý Haarův příznak lze vyjádřit jako kombinaci „černých“ a „bílých“ obdélníků, jak je možné vidět na obrázku 6. Hodnota Haarova příznaku se vypočítá jako rozdíl sum hodnot pixelů tmavé (černý obdélníkový region) a světlé (bílý obdélníkový region) části obrazu. Výsledná suma je použita k nalezení rozdílu mezi regiony. Rozdíly mohou být použity pro klasifikaci podoken v obraze.



Obrázek 6: Obrázek A, B zobrazuje Hranový příznak, C zobrazuje Čárový příznak, D zobrazuje Diagonální příznak [6]

3.3 AdaBoost

AdaBoost (Adaptive Boosting) vychází ze strojového učení zvané boosting. Může být použit s mnoha dalšími typy učení algoritmu pro zlepšení jejich výkonu. Jedná se o klasifikační algoritmus, který je schopen generovat silný klasifikátor z kombinací slabých klasifikátorů (obrázek 7), jejichž přesnost je jen o něco větší než 50%. Slabý klasifikátor je matematicky popsán jako:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{jestliže } pf(x) > p\theta \\ 0 & \text{jinak} \end{cases}$$

Níže je uveden stručný popis učení algoritmu AdaBoost

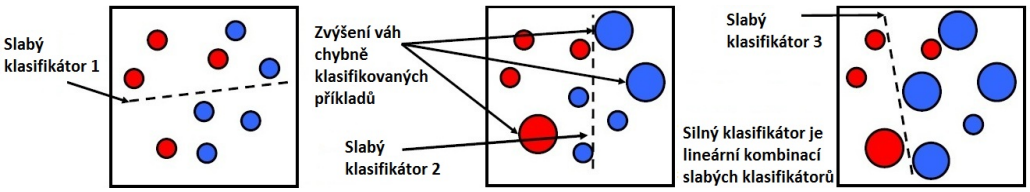
- Vstup: $(x_1, y_1), ..., (x_n, y_n)$,kde $y_i = 0, 1$ značí pozitivní nebo negativní příklady
- Inicializace vah: $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ pro $y_i = 0$, kde m a l je počet negativ a pozitiv
- Cyklus pro t = 1,...,T

1. Normalizace vah:
$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
2. Výběr slabého klasifikátoru s nejmenší chybou klasifikace při daných váhách
$$e_i = min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$
3. Výběr klasifikátoru h_t s nejnižší chybou e_t
4. Aktualizace váhy:
$$w_{t+1,i} = w_t, i\beta_t^{1-e_t}$$

kde $e_i = 0$ je-li x_i klasifikován správně, jinak $e_i = 1$, a $\beta_t = \frac{e_t}{1-e_t}$

- Silný klasifikátor je dán vztahem:
$$C(x) = \begin{cases} 1 & \text{jestliže } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{jinak} \end{cases}$$

kde
$$\alpha_t = \log \frac{1}{\beta_t}$$



Obrázek 7: Grafický popis práce AdaBoost

3.4 Kaskádový klasifikátor

Účelem kaskádového klasifikátoru je co nejrychlejší nalezení požadovaného objektu. Kaskáda je jakási struktura rozhodovacího stromu rozdělena do několika stupňů. Tyto jednotlivé stupně jsou sestaveny z trénovaných klasifikátorů nebo jejich kombinací. Jakmile jeden z těchto stupňů označí podokno vstupního obrazu za negativní, je toto podokno okamžitě odstraněno. Je-li podokno označeno jako pozitivní, pokračuje k dalšímu stupni. První stupeň kaskády odstraní velký počet negativních podoken s velmi krátkou dobou zpracování. Následující stupně kaskády odstraní další negativní podokna, narůstá však doba jejich zpracování. Po několika stupních kaskády je počet podoken výrazně snížen. Projde-li podokno přes všechny stupně kaskády, je považováno za hledaný objekt.

4 Houghova transformace

Houghova transformace se používá k detekci určitých tvarů v obraze. Princip této metody byl poprvé publikován P.V.C Houghem [7] v roce 1959. Jelikož metoda vyžaduje, aby požadované vlastnosti byly specifikovány v parametrické podobě, používá se Houghova transformace k detekci pravidelných křivek, jako jsou přímky, kružnice, elipsy. Aby byla pravděpodobnost nalezení hledané struktury v obraze co nejvyšší a náročnost výpočtů celé operace co nejnižší, je důležité převést vstupní obraz na obraz ve stupních šedi a předzpracovat hranovým detektorem. Hlavní výhodou Houghovy transformace je tolerance mezer v hledané struktuře, či postižení velkým šumem. Nevýhodou je vyšší časová náročnost algoritmu při hledaní složitých tvarů.

Houghovu transformaci je možné definovat jako hledání takové podmnožiny bodů, která nejvíce odpovídá části přímky. Každý bod přímky je dán souřadnicemi (x, y).

$$r = x \cos \varphi + y \sin \varphi$$

kde r je délka normály přímky od počátku souřadnic a φ je úhel mezi normálou a osou x. Pro bod A přejde rovnice do tvaru

$$r = x_1 \cos \varphi + y_1 \sin \varphi$$

Obdobný algoritmus lze použít pro hledání kružnic v obraze. Následující vztah vyjadřuje rovnici kružnice v polárních souřadnicích

$$(x - a)^2 + (y - b)^2 = R_2$$

Každý bod nacházející se na kružnici se středem v souřadnicích (a, b) a poloměrem R lze definovat vztahem:

$$x = a + R \cos \varphi$$

$$y = b + R \sin \varphi$$

Jsou - li známy souřadnice bodu s parametry x, y, který leží na kružnici s daným poloměrem R, je vypočtena hodnota parametru a a b dle vztahu:

$$a = x - R \cos \varphi$$

$$b = y - R \sin \varphi$$

Ty body, které budou mít stejnou hodnotu parametru a a b, leží na dané kružnici.

5 Detekce hran

Detekce hran patří spolu s potlačením šumu, jasovým a geometrickým transformacím k základním operacím předzpracování obrazu. Hrana se nachází v místech, kde dochází k velké změně jasových hodnot. Samotné detektory hran jsou sestaveny právě tak, aby byly citlivé v detekci libovolné změny jasových hodnot. Aby se zabránilo falešným detekcím hran, je potřeba potlačení šumu před samotnou hranovou filtrací. Po potlačení šumu v obraze je dalším krokem detekce hran. Jeden z nejlepších výsledků v oblasti detekce hran dosahuje Cannyho hranový detektor.

5.1 Cannyho hranový detektor

Poprvé byl tento detektor publikován Johnem Cannym [8] v roce 1986. Jedná se o jeden z nejčastěji používaných hranových detektorů. Cannyho hranový detektor umožňuje detekovat hrany s vysokou přesností a nízkou chybovostí. Výstup tohoto detektoru je dvouúrovňový obraz, kde 0 představuje pozadí a 1 hrany.
Cannyho hranový detektor definuje 3 klíčové požadavky pro optimální detekci hran:

- nízká chybovost, všechny hrany by měly být nalezeny a neměly by se detekovat místa, které hranami nejsou
- hranový bod je detekován s maximální přesností
- hranový bod je detekován pouze jednou

Algoritmus tohoto detektoru může být popsán v těchto bodech:

1. Eliminace šumu.
2. Výpočet velikosti a směrů gradientu.
3. Pro každý pixel nalezení derivace ve směru gradientu pomocí „optimální“ masky spojující vyhlazení a derivaci.
4. Nalezení lokálního maxima těchto derivací.
5. Hranové body získané prahováním s hysterezí.

Mechanismus prahování s hysterezí aplikuje dvě prahové hodnoty, vyšší T_H a nižší T_L. Pokud je hranová hodnota vyšší než práh T_H, je tato hodnota okamžitě uznána jako hrana. Je-li však prahová hodnota nižší než práh T_L, je trvale zamítnuta. Pokud se hranová hodnota nachází v rozmezí mezi T_H a T_L, je považována za hranu v případě, že je sousedící hranová hodnota uznána jako hrana.

6 Implementace systému

Implementace systému probíhala v prostředí Microsoft Visual Studio 2012, v jazyce C++ s využitím knihovny OpenCV [9].

OpenCV je knihovna programových funkcí zaměřených především na počítačové vidění v reálném čase, původně vyvinutá společností Intel. Knihovna je napsána v jazycích C/C++ a v rámci open-source licencí BSD zdarma pro akademické i komerční využití. OpenCV podporuje jak desktopové operační systémy Windows, Linux, OS X, FreeBSD, NetBSD, OpenBSD, tak i mobilní verze pro systémy Android, iOS, Maemo, BlackBerry 10.

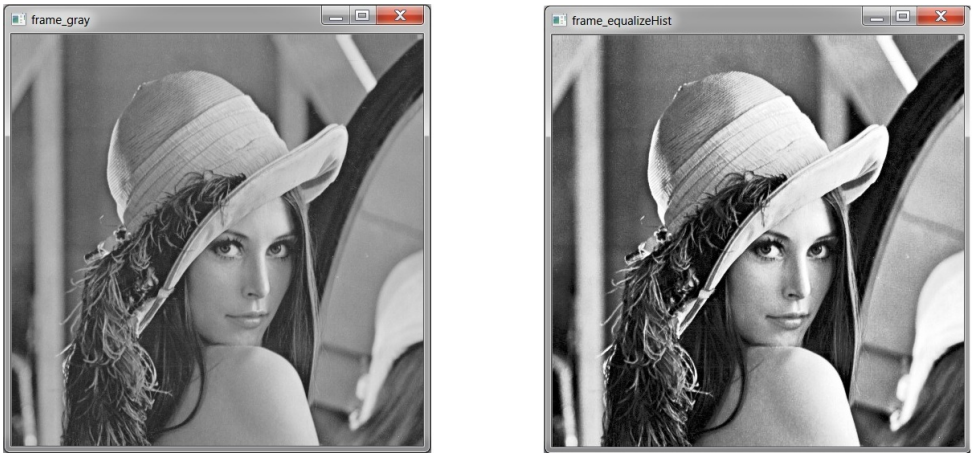
6.1 Předzpracování obrazu

Než dojde k samotné detekci, je potřeba převést vstupní barevný obraz na obraz ve stupních šedi, známý také jako šedotónový obraz. V tomto obraze je každý pixel vyjádřen hodnotou intenzity šedi (0-255 pro osmibitový obraz). Tuto konverzi barevného obrazu do obrazu ve stupních šedi lze vyjádřit vztahem:

$$Y \leftarrow 0,299R + 0,587G + 0,114B$$

kde Y je intenzita šedi a R, G, B jsou spektrální složky barevného obrazu. Tento výpočet je proveden pro každý pixel obrazu. V OpenCV je tato funkce implementována jako cvtColor s parametrem CV_RGB2GRAY (obrázek 8).

Dalším krokem zpracování je ekvalizace (vyrovnání) histogramu. To je možné chápat jako vyrovnání jasových hodnot v obraze. Výsledkem použití této funkce je zvýšení kontrastu obrazu. V OpenCV je tato funkce implementována jako *equalizeHist* (obrázek 8).



Obrázek 8: Levý snímek: šedotónový obraz, pravý snímek: obraz po ekvalizaci

6.2 Detekce tváře

Detekce tváře je jeden z hlavních kroků v systému pro detekci únavy řidiče. Pokud není systém schopen detekovat tvář ve vstupním obraze, jsou další operace spojené se zpracováním obrazu zbytečné a neprovádí se. Pokud však systém spolehlivě tvář detekuje, použije se její poloha ke zmenšení oblasti pro zpracování obrazu a další operace již probíhají jen v oblasti tváře. V tomto případě byl použit k detekci tváře klasifikátor *haarcascade_frontalface_alt.xml*, který je přikládán ke knihovně OpenCV.

6.3 Detekce očí

Je-li detekována tvář, je dalším krokem vymezení oblasti, kde lze detekovat oči. Z faktu, že se oči nacházejí v horní polovině tváře, bude snížena výška oblasti detekované tváře o 50%. Z této oblasti lze pak snadněji získat samotné oblasti pro pravé a levé oko. Pro úlohu detekce očí lze využít stejných metod jako při detekci tváře. Jako nejvhodnější se projevil ke knihovně přikládaný klasifikátor *haarcascade_righteye_2splits.xml* a *haarcascade_lefteye_2splits.xml* pro pravé a levé oko. Výsledek očního detektoru obsahuje kromě samotného oka také obočí, které může představovat zdroj falešných detekcí. Z toho důvodu bude snížena celková výška oblasti detekce oka o 50%.

6.4 Detekce mrknutí očí

Existuje několik metod, jak lze detekovat mrknutí lidského oka. Tato práce podrobně rozebere tyto tři vybrané metody.

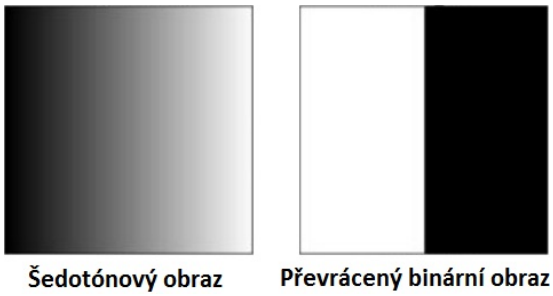
1. **Detekce zorničky pomocí prahování:**
   Pro zjištění stavu, zda je oko otevřené nebo zavřené, lze využít faktu, že zornička představuje nejtmavší bod celého oka. Pokud tento bod není v obraze nalezen, je možné předpokládat, že je oko zavřené.

2. **Detekce kružnice v lidském oku pomocí Houghovy transformace:**
   Jedná se o principiálně obdobnou metodu jako v předchozím případě, tentokrát se však pracuje s oční duhovkou, která tvoří kružnici. Tuto kružnici je možné detekovat jen v případě, že je oko otevřené. Detekce kružnice bude provedena pomoci Houghovy transformace a vhodným předzpracováním obrazu.

3. **Detekce mrknutí očí pomocí natrénovaného klasifikátoru**
   Stejně tak jako u detekce tváře a očí lze použít tentýž princip i k detekci samotného mrkání. Jelikož knihovna OpenCV neobsahuje takový klasifikátor, který by byl schopný detekovat toto mrkání, bude za pomoci trénování takový kaskádový klasifikátor vytvořen. K trénování takového klasifikátoru je zapotřebí dvou typů vzorků, a to pozitivní a negativní. Pozitivní vzorky obsahují sbírku snímků zavřených očí, negativní vzorky sbírku otevřených očí.

6.4.1 Detekce zorničky pomocí prahování

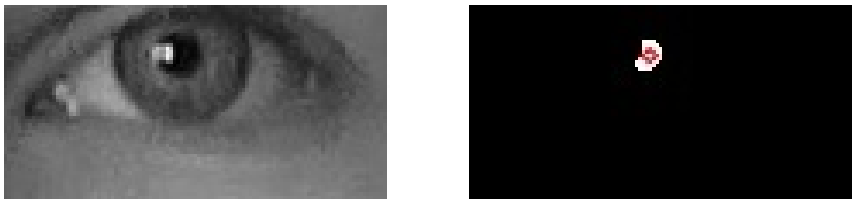
Prahování je nejjednodušší způsob segmentace. Odděluje oblasti obrazu, které odpovídají objektům, které chceme analyzovat. Toto oddělení je založeno na kolísání intenzity mezi pixely popředí a pixely pozadí. Všechny hodnoty intenzity pixelu nižší než prahová hodnota odpovídají pozadí, hodnoty vyšší odpovídají popředí. OpenCV poskytuje různé typy prahování, v této práci byl použit typ, jehož výstupem byl převrácený binární obraz (obrázek 9).
Příklad použití:
threshold(image, image, 35, 255, CV_THRESH_BINARY_INV);



Obrázek 9: Ukázka použití prahování

Po prahování je nutné detekovat, zda obraz obsahuje nejtmavší oblast (zorničku). To je provedeno pomocí Blob detektoru.

Blob (skvrna, kapka) je skupina pixelů, které mají stejnou hodnotu šedé. Cílem detekce Blob je identifikovat a označit tyto skupiny. Parametry pro *SimpleBlobDetector* jde nastavit filtr pro typ „skvrny“, který je potřeba nalézt. Samotné „skvrny“ se mohou filtrovat podle barvy, velikosti, tvaru. Obrázek 10 zobrazuje vstupní obraz oka a obraz oka po použití prahování. Červená kružnice znázorňuje, že došlo k detekci zorničky.

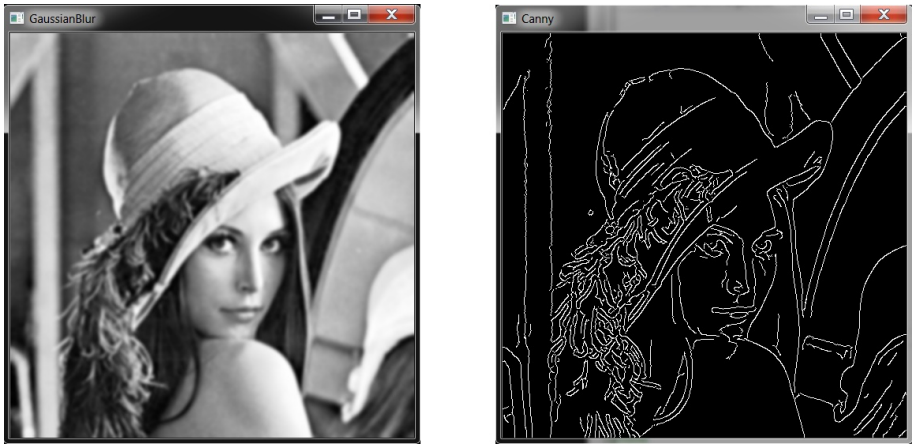


Obrázek 10: Ukázka detekce zorničky pomocí prahování

6.4.2 Detekce kružnice v lidském oku pomocí Houghovy transformace

Další z metod, jak lze realizovat detekci mrknutí oka, je použití Houghovy transformace pro nalezení kružnic. Aby byla co nejefektivněji a nejpřesněji detekována duhovka, jakožto kružnice v lidském oku pomocí této metody, je nutné si vstupní obraz dále předzpracovat. To bude provedeno pomocí Gaussova filtru (obrázek 11), který ze vstupního obrazu odstraní rušivý šum

a dojde tak k vyhlazení obrazu. V OpenCV je tento filtr implementován jako funkce *GaussianBlur*. Součástí předzpracování je i použití morfologických operací, v tomto případě se jedná o operace eroze a dilatace. Tyto operace jsou použity pro izolaci jednotlivých prvků a spojování nesourodých prvků, pro nalezení nerovností intenzit nebo zaplnění děr v obraze. V OpenCV jsou tyto operace implementovány jako *Erode* a *Dilate*. Následuje detekce hran pomocí Cannyho hranového detektoru (obrázek 11), který je implementován jako funkce *Canny*.



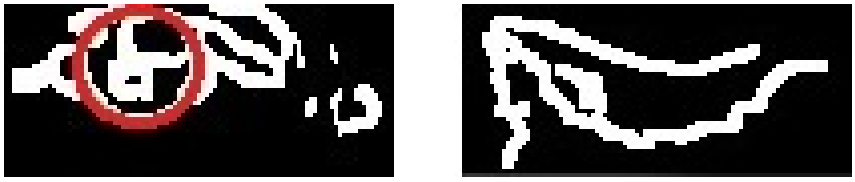
Obrázek 11: Levý snímek: obraz po použití Gaussova filtru, pravý snímek: obraz po aplikaci Cannyho hranového detektoru

Nyní je možné přejít k samotné detekci kružnic. To je provedeno použitím funkce *HoughCircles*. Parametry funkce **HoughCircles**:

- **image** - Vstupní obraz ve stupních šedi.
- **circles** - Výstup, jakožto vektor nalezených kružnic. Každý vektor je dán třemi elementy $(x, y, poloměr)$.
- **method** - Metoda použitá pro detekci, v současné době jediná: *CV_HOUGH_GRADIENT*
- **dp** - Inverzní poměr rozlišení akumulátoru vůči rozlišení obrazu. Pokud $dp = 1$, má akumulátor stejné rozlišení jako vstupní obraz, pokud $dp = 2$, má akumulátor z poloviny tak velkou šířku a výšku.
- **minDist** - Minimální vzdálenost mezi středy detekovaných kružnic.
- **param1** - Vyšší prahová hodnota předaná z Cannyho hranového detektoru.
- **param2** - Prahová hodnota pro středy kružnic. Čím menší hodnota je, tím více je detekováno falešných kružnic.
- **minRadius** - Minimální poloměr kružnice.
- **maxRadius** - Maximální poloměr kružnice.

Příklad použití:
HoughCircles(eyeROICanny, circles, CV_HOUGH_GRADIENT, 1, 10, 100, 20, 10, 40);

Obrázek 12 zobrazuje obraz oka po předzpracování obrazu Gaussovým filtrem a Cannyho hranovým detektorem. Červená kružnice znázorňuje, že došlo k detekci duhovky v otevřeném oku.



Obrázek 12: Ukázka detekce kružnice, levý snímek - otevřené oko, pravý snímek - zavřené oko

6.4.3 Detekce mrknutí očí pomocí natrénovaného klasifikátoru

Knihovna OpenCV obsahuje již několik natrénovaných klasifikátorů. Jedná se především o klasifikátory detekující lidské postavy, tváře a obličejové části, jako jsou oči, nos, uši, ústa. Chceme-li však v obraze detekovat jakýkoli jiný objekt, je potřeba si natrénovat vlastní kaskádový klasifikátor. K tomu účelu obsahuje knihovna aplikace *opencv_createsamples* a *opencv_traincascade*, které budou dále popsány. Jak již bylo dříve zmíněno, je k natrénování nezbytné vytvoření dvou typů vzorků - pozitivní a negativní.

6.4.3.1 Pozitivní vzorky

Za pozitivní vzorky se považují snímky, které obsahují námi hledaný objekt, v tomto případě se jedná o snímky zavřených očí (obrázek 13). Tyto pozitivní vzorky jsou vyjmenovány v textovém souboru, kde každý řádek obsahuje název a cestu k pozitivnímu vzorku (obrázek 14). Dále následuje počet hledaných objektů ve snímku a souřadnice popisující obdélník (x, y, šířka, výška). Všechny tyto vzorky musí být stejné velikosti.

Obrázek 13: Ukázka pozitivních vzorků

a)

/pozitivni_vzorky

pozitiv(1).jpg

pozitiv(2).jpg

...

pozitiva.txt

b)

pozitivni_vzorky/pozitiv(1).jpg 1 10 5 24 29

pozitivni_vzorky/pozitiv(2).jpg 2 12 3 20 30 40 35 26 28

...

Obrázek 14: a) Adresářová struktura, b) Struktura textového souboru pozitiva.txt

Snímek pozitiv(1).jpg obsahuje jeden hledaný objekt s následujícími souřadnicemi (10, 5, 24, 29), snímek pozitiv(2).jpg obsahuje dva hledané objekty.

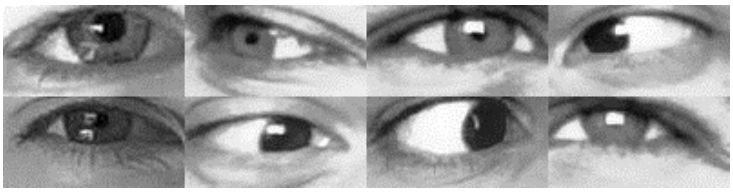
Pomocí aplikace opencv_createsamples je z textového souboru vytvořen soubor typu .vec. Parametry **opencv_createsamples**:

- **-info** <název_souboru_pozitiv>
  Soubor, který obsahuje popis sady pozitivních vzorků.
- **-vec** <název_souboru_vec>
  Výstupní vektorový soubor.
- **-w** <šířka_pozitivních_vzorků>
- **-h** <výška_pozitivních_vzorků>
- **-num** <počet_pozitivních_vzorků>

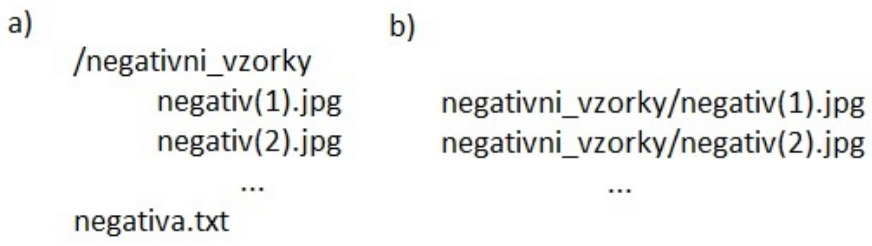
Příklad použití:
opencv_createsamples -info pozitiva.txt -vec pozitiva.vec -w 20 -h 10 -num 2000

6.4.3.2 Negativní vzorky

Negativní vzorky představují libovolné snímky, které neobsahují hledaný objekt, v tomto případě se jedná o snímky otevřených očí (obrázek 15). Tyto negativní vzorky jsou vyjmenovány v textovém souboru, kde každý řádek obsahuje název a cestu k negativnímu vzorku (obrázek 16). Negativní vzorky mohou být různých velikostí, měly by však být větší než vzorky pozitivní.



Obrázek 15: Ukázka negativních vzorků



Obrázek 16: a) Adrešářová struktura, b) Struktura textového souboru negativa.txt

6.4.3.3 Trénování kaskádového klasifikátoru

Posledním krokem je trénování klasifikátoru za pomoci aplikace *opencv_traincascade*. Tato aplikace podporuje jak LBP, tak i Haarovy příznaky. LBP je vzhledem k Haarovým příznakům mnohonásobně rychlejší, přičemž výsledný klasifikátor může dosahovat téměř stejné kvality jako při použití Haara. Pokud jde o kvalitu výsledného klasifikátoru, záleží především na kvalitě trénovacích vzorků a správně zvolených parametrech tréninku. Výstupem aplikace je klasifikátor cascade.xml, který se nachází ve složce, která byla předána jako parametr -data. Ostatní soubory .xml jsou v této složce vytvořeny pro případ přerušení tréninku, takže je lze po ukončení tréninku odstranit.
Parametry **opencv_traincascade**:

- **-data** <název_složky_kaskády>
  Složka, do které bude uložen natrénovaný klasifikátor.
- **-vec** <název_souboru_vec>
  Vektorový soubor vytvořený aplikací opencv_createsamples, který obsahuje pozitivní vzorky pro trénink.
- **-bg** <název_souboru_negativ> Soubor, který obsahuje popis sady negativních vzorků.
- **-numPos** <počet_pozitivních_vzorků>
- **-numNeg** <počet_negativních_vzorků>
- **-numStages** <počet_kaskádových_stupňů>
- **-precalsValBufSize** <velikost_paměti_pro_hodnoty_příznaku>
  Velikost vyrovnávací paměti v Mb pro předem vypočtené hodnoty příznaku
- **-precalsIdxBufSize** <velikost_paměti_pro_indexy_příznaku>
  Velikost vyrovnávací paměti v Mb pro předem vypočtené indexy příznaku, větší paměť má za následek zrychlení procesu tréninku.

- **-acceptanceRatioBreakValue**
  Tento argument určuje, jak přesné má být trénování klasifikátoru a kdy ho zastavit. V ideálním případě by hodnota neměla překročit 10e-5, jinak by mohlo dojít k přetrénování, což by mělo za následek špatnou nebo žádnou detekci.

- **-featureType** <HAAR(default), LBP>
  Typy příznaků: HAAR - Haarovy příznaky, LBP - lokální binární vzory

- **-w** <šířka_vzorku>

- **-h** <výška_vzorku>
  Šířka a výška vzorku musí být stejná jako při použití aplikace opencv_createsamples.

- **-minHitRate** <minimální_přesnost>
  Minimální požadovaná míra přesnosti pro každý stupeň klasifikace. Celkovou míru přesnosti lze odhadnout jako (minimální_přesnost ^ počet_kaskádových_stupňů).

- **-maxFalseAlarmRate** <maximální_počet_falešných_poplachů>
  Maximální požadovaný počet falešných poplachů pro každý stupeň klasifikace. Celkový počet falešných poplachů lze odhadnout jako (maximální_počet_falešných_poplachů ^ počet_kaskádových_stupňů.

Příklad použití:
opencv_traincascade -data cascade -vec pozitiva.vec -bg negativy.txt -numPos 1852 -numNeg 3413 -numStages 18 -precalsValBufSize 2048 -precalsIdxBufSize 2048 -maxFalseAlarmRate 10e-5 -featureType LBP -w 20 -h 10 -minHitRate 0.999 -maxFalseAlarmRate 0.5

6.4.3.4 Detekce pracující s natrénovaným klasifikátorem

OpenCV obsahuje k detekci pomocí natrénovaných klasifikátorů funkci *detectMultiScale*. Tato funkce detekuje objekty různých velikostí. Zjištěné objekty jsou vráceny jako vektor obdélníků. Tato funkce je použita jak k samotné detekci mrkání, tak i k detekci tváře a očí.
Parametry funkce **detectMultiScale**:

- **image** - Obraz, ve kterém chceme detekovat hledaný objekt.
- **objects** - Vektor obdélníků, kde každý obdélník obsahuje detekovaný objekt.
- **scaleFactor** - Tento parametr určuje, o kolik je snížena velikost obrazu v každém obrazovém měřítku.
- **minNeighbors** - Tento parametr určuje minimální počet sousedních objektů, které by měl hledaný objekt mít.
- **flags** - Stejný význam jako předešlý parametr. Využívá se u starých kaskád, u nových nikoli.
- **minSize** - Minimální možná velikost hledaného objektu. Objekt menší než minSize je ignorován.
- **maxSize** - Maximální možná velikost hledaného objektu. Objekt větší než maxSize je ignorován.

Příklad použití:
cascade.detectMultiScale(fr_gray, faces, 1.3, 3, 0|CASCADE_SCALE_IMAGE, Size(60, 60));

7 Testování

Systém byl primárně vyvíjen a testován na operačním systému Windows 7 64bit, procesor Intel Core i5-2400, 8GB RAM. Testování metod proběhlo na vlastně vytvořené databází snímku, které byly extrahovány a náhodně vygenerovány z osobních videí. Na videích se vyskytovali 4 různé osoby ve 3 různých prostředích. Snímky byly testovány s rozlišením 1920x1080 a obsahovaly tvář s otevřenýma nebo zavřenýma očima.
Výpočet přesnosti:

$$Přesnost[\%] = \frac{TP + TN}{TP + TN + FP + FN} * 100$$

kde TP je počet správně detekovaných snímků se zavřenýma očima, TN počet snímků, kde systém správně detekoval otevřené oči, FP počet snímků, kde systém detekoval zavřené oči jako otevřené a FN počet snímků, kde systém detekoval otevřené oči jako zavřené. Výsledky testování všech 3 použitých metod jsou postupně znázorněny v obrázcích 17, 18 a 19.

Prahování	Osoba 1	Osoba2	Osoba 3	Osoba 4
Počet snímků	44	48	30	36
TP	17	17	10	8
TN	17	16	9	17
FP	5	5	1	0
FN	5	10	10	11
Přesnost [%]	77,27	68,75	63,33	69,44
Průměrná přesnost [%]	69,70			

Obrázek 17: Detekce zorničky pomocí prahování

V případě detekce zorničky při použití prahování hraje velkou roli odraz v očích. Pokud je odraz v oku příliš velký, zastíní zorničku, což znemožní jakoukoli její detekci. Dalším problémem jsou tmavé řasy, které jsou v mnoha případech zdrojem falešných detekcí (obrázek 25). Výhodou této metody je, že dokáže detekovat mrknutí i v případě, kdy snímaná osoba změnila směr pohledu očí do stran.

Houghova transformace	Osoba 1	Osoba2	Osoba 3	Osoba 4
Počet snímků	44	48	30	36
TP	17	14	10	8
TN	15	18	8	25
FP	5	8	1	0
FN	7	8	11	3
Přesnost [%]	72,73	66,67	60,00	91,67
Průměrná přesnost [%]	72,77			

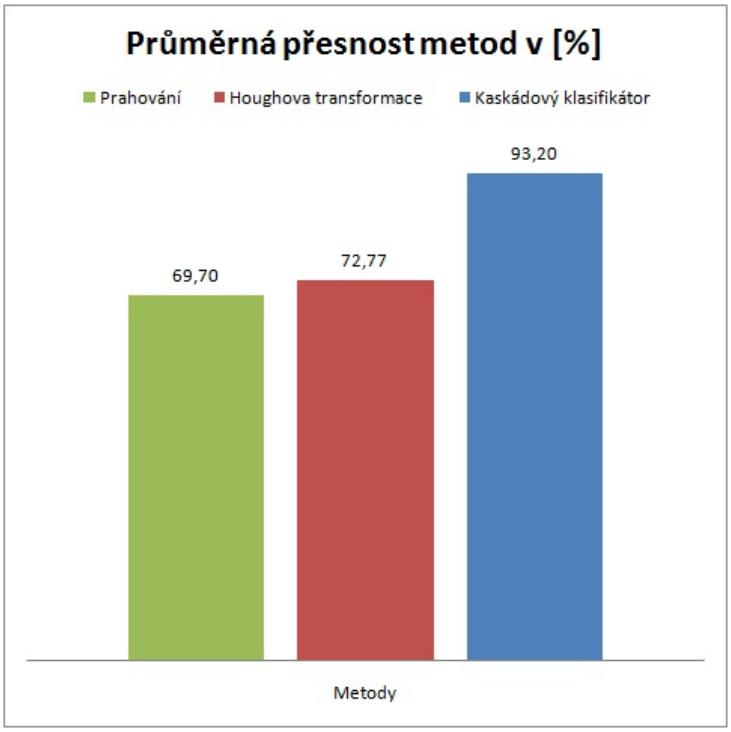
Obrázek 18: Detekce mrknutí pomocí Houghovy transformace

Hlavním problémem metody využívající Houghovy transformace k nalezení kružnice v oku je neschopnost detekovat duhovku, jakožto kružnici v případě, kdy snímaná osoba výrazně změnila směr pohledu očí do stran nebo očima mžourá, nedostatek je vyobrazen v obrázku 25.

Natrénovaný klasifikátor	Osoba 1	Osoba2	Osoba 3	Osoba 4
Počet snímků	44	48	30	36
TP	21	18	11	8
TN	18	24	18	28
FP	1	4	0	0
FN	4	2	1	0
Přesnost [%]	88,64	87,50	96,67	100,00
Průměrná přesnost [%]	93,20			

Obrázek 19: Detekce mrknutí pomocí natrénovaného klasifikátoru

Jak již bylo zmíněno výše, úspěšnost detekce mrknutí pomocí natrénovaného klasifikátoru záleží na kvalitě trénovacích vzorků, správně zvolených parametrech tréninku a detekce. Čím větší bude sbírka a různorodost trénovacích vzorků, tím bude detekce mrkání očí přesnější.
V tomto případě množina představovala přes 2000 pozitivních a téměř 3500 negativních vzorků, což pro testování úspěšnosti detekce považuji za dostačující.



Obrázek 20: Graf znázorňující přesnost metod

Průměrná přesnost metod je blíže porovnána a znázorněna grafem v obrázku 20. Jak je z grafu patrné, nejlepšího výsledku detekce dosahovala metoda využívající natrénovaného klasifikátoru s přesností přes 93%. Velmi podobných výsledků pak dosahovaly metody založené na detekci zorničky pomocí prahování a detekci mrknutí pomocí Houghovy transformace, kdy obě tyto metody pracovaly s přesností okolo 70%.

Průměrný čas detekce mrknutí ve snímku	
Prahování	250,03 ms
Houghova transformace	253,36 ms
Natrénovaný klasifikátor	259,24 ms

Obrázek 21: Časová náročnost metod pro detekci mrknutí

Metody detekce mrknutí jsou porovnány i z hlediska časové náročnosti (obrázek 21). Jednotlivé časy byly získány při testování vzorků a zahrnují jak detekci mrknutí, tak i detekci tváře a očí. Z obrázku 21 je patrné, že detekce mrknutí pomocí natrénovaného klasifikátoru je z časového hlediska nejnáročnější.

Úspěšnou detekci mrknutí na testovacích snímcích při použití prahování k nalezení zorničky zobrazuje obrázek 22. Červená kružnice v tomto případě znázorňuje, že došlo k detekci zorničky.

Obrázek 22: Ukázka úspěšné detekce mrknutí očí pomocí prahování

Úspěšnou detekci mrknutí na testovacích snímcích při použití Houghovy transformace zobrazuje obrázek 23. Zelená kružnice v tomto případě znázorňuje, že došlo k detekci kružnice v oku.

Obrázek 23: Ukázka úspěšné detekce mrknutí očí pomocí Houghovy transformace

Úspěšnou detekci mrknutí na testovacích snímcích při použití natrénovaného klasifikátoru zobrazuje obrázek 24. Žlutá kružnice v tomto případě znázorňuje, že došlo k detekci zavřeného oka.

Obrázek 24: Ukázka úspěšné detekce mrknutí očí pomocí natrénovaného klasifikátoru

Obrázek 25: Ukázka špatné detekce mrknutí očí

Levé snímky zobrazují nedostatek detekce při použití Houghovy trasformace, v tomto případě se jedná o mžourání očí. Pravé snímky zobrazují nedostatky detekce při prahování, tmavé řasy jako zdroj falešné detekce a velký odraz v oku, který v mnoha případech znemožní detekci zorničky.

8 Závěr

Cílem této práce bylo vytvoření aplikace, která by byla schopna detekce mrknutí oka v obrazech. Aplikace byla implementována 3 různými metodami.

V teoretické části byla popsána základní funkčnost metod, které byly použity při implementaci aplikace. Jedná se o metody Local Binary Patterns, Viola-Jones, Houghova transformace a funkce spojené s předzpracováním obrazu.

V praktické části byla popsána implementace těchto metod a funkce s nimi spojené, včetně popisu tréninku kaskádového klasifikátoru. Následně došlo k jejich testování a porovnání. Testování všech tří metod detekce proběhlo na stejné množině testovacích vzorků. Nejlepších výsledků dosahovala metoda využívající natrénovaného kaskádového klasifikátoru, která byla z časového hlediska nejnáročnější.

Ačkoli každá z metod pracuje na jiném principu detekce, pro každou z metod hraje důležitou roli intenzita osvětlení, kvalita snímacího zařízení a úhel pohledu snímaní. Předzpracování obrazu má rovněž velký vliv na detekci. Nelze zvolit univerzální nastavení pro všechny typy obrazu, protože nastavení předzpracování je pro každý obraz individuální. V této práci byla vytvořena konzolová aplikace, která je schopná detekce tváře, očí a mrkání. Aplikace je určena pro testování, nikoli pro použití v reálném životě.

Aplikaci by bylo možné dále rozšířit o uživatelské rozhraní, kde by bylo možné nastavit parametry vhodné pro detekci v danou chvíli.

Literatura

[1] OJALA, Timo; PIETIKÄINEN, Matti; MÄENPÄÄ, Topi. *Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns* [online]. [cit. 9.6.2016]. Dostupný na WWW: http://www.outex.oulu.fi/publications/pami_02_opm.pdf

[2] WAGNER, Philipp. *Local Binary Patterns* [online]. [cit. 21.6.2016]. Dostupný na WWW: http://bytefish.de/blog/local_binary_patterns/

[3] PIETIKÄINEN, Matti; HEIKKILÄ, Janne. *Image and Video Description with Local Binary Pattern Variants* [online]. [cit. 9.6.2016]. Dostupný na WWW: http://www.ee.oulu.fi/research/imag/mvg/files/pdf/CVPR-tutorial-final.pdf

[4] HÄFNER, M.; LIEDLGRUBER, M.; UHL, A. a kol. *Color treatment in endoscopic image classification using multi-scale local color vector patterns* [online]. [cit. 22.6.2016]. Dostupný na WWW: http://www.medicalimageanalysisjournal.com/article/S1361-8415(11)00056-9/pdf

[5] VIOLA, Paul; JONES, Michael J. *Robust Real-Time Face Detection* [online]. [cit. 9.6.2016]. Dostupný na WWW: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

[6] *The Viola/Jones Face Detector (2001)* [online]. [cit. 9.6.2016]. Dostupný na WWW: http://www.cs.ubc.ca/ lowe/425/slides/13-ViolaJones.pdf

[7] VLACH, Jaroslav. *Hledání úseček a kružnic s využitím Houghovy transformace při zpracování obrazu v LabView* [online]. [cit. 22.6.2016]. Dostupný na WWW: http:automa.cz/res/pdf/42983.pdf

[8] *Canny Edge Detection* [online]. [cit. 9.6.2016]. Dostupný na WWW: http://www.cse.iitd.ernet.in/ pkalra/csl783/canny.pdf

[9] *OpenCV* [online]. [cit. 23.6.2016]. Dostupný na WWW: https://en.wikipedia.org/wiki/OpenCV

A Obsah CD

Přiložené CD obsahuje tyto soubory:

- \BP - HOL0229\ - adresář aplikace
- HOL0229.pdf - text bakalářské práce ve formátu PDF
- adresarova_struktura.txt - textový soubor s adresářovou strukturou aplikace